EECS-214 Spring 2016 Practice Quiz 1

Answer the following questions as best you can. When writing code, you may not use any built-in sequence types other than arrays. In other words: if you need to use linked-lists, then you should build the lists yourself rather than assuming the presence of a pre-made linked list class, such as the C# LinkedList<T> class.

Note: the real test will not be this long.

Question 1

Consider the following pseudocode:

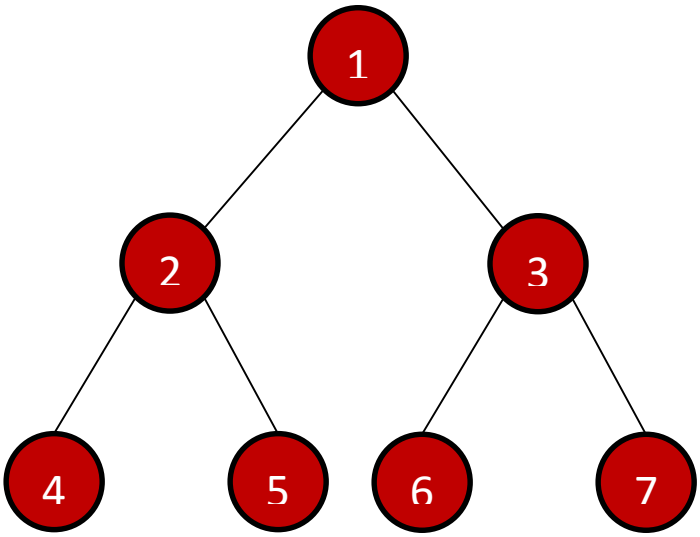
```
PrintSorted(list of numbers) {
    make a new, empty, binary search tree
    for each number in the list
        add it to the binary search tree
    node = minimum node of the BST
    while node != null {
        print node.key
        node = successor(node)
    }
}
```

A. What is the worst case time complexity (i.e. O(1), O(n), etc.) of this algorithm if we use a normal (i.e. not self-balancing) binary search tree?

B. What if we use a red/black tree instead of a normal binary tree?

Question 2

Give the order in which the nodes of the tree below would be printed by an in-order traversal



Question 3

Give an algorithm for walking the tree above that, if run on the tree above, would walk the nodes in the order 1, 2, 3, 4, 5, etc. For this question, you may assume that stack and queue data types have already been defined and that you can use them for your answer on this question without having to write code for them.

Question 4

Consider the following linked-list queue implementation:

```
class Cell {
   Cell next;
   int value;
}

class Queue {
   Cell head;
   Cell Tail() {
       Cell c = head;
       while (c.next != null) c = c.next;
      return c;
   }

   void Enqueue(value) {
      Cell t = new Cell();
       t.value = value;
       t.next = null;


       if (head==null)
           head = t;
      else
            Tail().next = t;
   }
}
```

This implementation has the unfortunate consequence that building a queue with n items takes $O(n^2)$ time (assuming we don't dequeue anything in-between). Explain how to fix this so that it takes only O(n) time.

Question 5

Write a class definition for a tree node for a tree. Assume the tree can have an arbitrary branching factor (i.e. no limit on the number of children). You need only give the code for the fields; you do not need to define any methods for the class.

Question 6

Given the following definition for a doubly-linked list:

```
class DLLCell {
   int key;
   DLLCell previous;
   DLLCell next;
}

class DLList {
   DLLCell head;    // first cell in the list
}
```

Write a procedure, AddAfter(DLList list, DLLCell c, int key) that adds a new cell containing *key* to the list *list*. It should add it immediately after the cell *c*, which you may assume is already in the list *list*. If *c* is null, you should add the cell at the beginning of the list.

Question 7

Here's a random algorithm that operates on an array. Yes, I know it doesn't do anything useful, but tell me what its time complexity is anyway (i.e. is it linear, quadratic, log n, or what?).

```
BlaBlaBla(array, start, end)
   if (start!=end)
     for each i between start and end
       sum += array[i]
     BlaBlaBla(array, start, (start+end)/2)
     BlaBlaBla(array, (start+end)/2, end)
```

Question 8

Draw a binary search tree for containing the numbers 1, 2, 3, 4, and 5. It need not be a balanced tree, it need only be a valid binary search tree.

Question 9

Give the algorithm for searching a binary search tree. What is its **worst case** time complexity in terms of the height of the tree? In terms of the number of items in the tree? Do not assume the tree is balanced.